Package: baritsu (via r-universe)

July 19, 2024

Title Wrappers for 'mlpack'

Version 0.0.2

Description A collection of wrappers for the 'mlpack' package that allows passing formula as their argument.

License MIT + file LICENSE

URL https://paithiov909.github.io/baritsu/

BugReports https://github.com/paithiov909/baritsu/issues

Depends parsnip

Imports generics, hardhat, mlpack, rlang, stats, tibble, utils

Suggests dplyr, modeldata, recipes, rsample, spelling, testthat (>= 3.0.0), workflows

Config/testthat/edition 3

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Repository https://paithiov909.r-universe.dev

RemoteUrl https://github.com/paithiov909/baritsu

RemoteRef HEAD

RemoteSha dec86eeec6800a173d4a72f12c4fce92367b45bc

Contents

`adaboost` *AdaBoost*

Description

A wrapper around `mlpack::adaboost()` that allows passing a formula.

Usage

```
adaboost(
  formula = NULL,
  data = NULL,
  epochs = 1000,
  tolerance = 1e-10,
  weak_learner = c("decision_stump", "perceptron"),
  x = NULL,
  y = NULL
)
```

Arguments

`formula`	A formula.
`data`	A data.frame.
`epochs`	The maximum number of boosting iterations to be run (0 will run until convergence.)
`tolerance`	The tolerance for change in values of the weighted error during training.
`weak_learner`	Weak learner to use. Either "decision_stump" or "perceptron".
`x`	Design matrix.
`y`	Response matrix.

Value

An object of class `baritsu_ab`.

See Also

`mlpack::adaboost()` `predict.baritsu_ab()`

`decision_trees` *Decision trees*

Description

A wrapper around `mlpack::decision_tree()` that allows passing a formula.

Usage

```
decision_trees(
  formula = NULL,
  data = NULL,
  tree_depth = 0,
  min_n = 20,
  minimum_gain_split = 1e-07,
  weights = NULL,
  x = NULL,
  y = NULL
)
```

Arguments

`formula`	A formula.
`data`	A data.frame.
`tree_depth`	Maximum depth of the tree.
`min_n`	Minimum number of data points in a leaf.
`minimum_gain_split`	Minimum gain required to split an internal node.
`weights`	Weights for each observation.
`x`	Design matrix.
`y`	Response matrix.

Details

To prevent masking of `parsnip::decision_tree()`, this function is named `decision_trees` (plural form!)

Value

An object of class `baritsu_dt`.

See Also

`mlpack::decision_tree()` `predict.baritsu_dt()`

`hoeffding_trees` *Hoeffding trees*

Description

A wrapper around `mlpack::hoeffding_tree()` that allows passing a formula.

Usage

```
hoeffding_trees(
  formula = NULL,
  data = NULL,
  confidence_factor = 0.95,
  sample_size = 10,
  max_samples = 5000,
  min_samples = 100,
  info_gain = FALSE,
  batch_mode = FALSE,
  numeric_split_strategy = c("binary", "domingos"),
  num_breaks = 10,
  observations_before_binning = 100,
  x = NULL,
  y = NULL
)
```

Arguments

`formula`	A formula.
`data`	A data.frame.
`confidence_factor`	Confidence before splitting (between 0 and 1).
`sample_size`	Number of passes to take over the dataset.
`max_samples`	Maximum number of samples before splitting.
`min_samples`	Minimum number of samples before splitting.
`info_gain`	Logical. If set, information gain is used instead of Gini impurity for calculating Hoeffding bounds.
`batch_mode`	Logical. If true, samples will be considered in batch instead of as a stream. This generally results in better trees but at the cost of memory usage and runtime.
`numeric_split_strategy`	The splitting strategy to use for numeric features.
`num_breaks`	If the "domingos" split strategy is used, this specifies the number of bins for each numeric split.
`observations_before_binning`	If the "domingos" split strategy is used, this specifies the number of samples observed before binning is performed.

x	Design matrix.
y	Response matrix.

Value

An object of class `baritsu_ht`.

See Also

`mlpack::hoeffding_tree()` `predict.baritsu_ht()`

`linear_regression` *Linear regression*

Description

A wrapper around `mlpack::linear_regression()` and `mlpack::lars()` that allows passing a formula.

Usage

```
linear_regression(
  formula = NULL,
  data = NULL,
  lambda1 = 0,
  lambda2 = 0,
  no_intercept = FALSE,
  no_normalize = FALSE,
  use_cholesky = FALSE,
  x = NULL,
  y = NULL
)
```

Arguments

formula	A formula.
data	A data.frame.
lambda1	Regularization parameter for L1-norm penalty.
lambda2	Regularization parameter for L2-norm penalty.
no_intercept	Logical; passed to `mlpack::lars()`.
no_normalize	Logical; passed to `mlpack::lars()`.
use_cholesky	Logical; passed to `mlpack::lars()`.
x	Design matrix.
y	Response matrix.

Details

When the lambda1 is 0, this function fallbacks to `mlpack::linear_regression()` for performance.

Value

An object of class `baritsu_lr`.

See Also

`mlpack::linear_regression()` `mlpack::lars()` `predict.baritsu_lr()`

---

linear_regression_bayesian

Bayesian linear regression

Description

A wrapper around `mlpack::bayesian_linear_regression()` that allows passing a formula.

Usage

```
linear_regression_bayesian(
  formula = NULL,
  data = NULL,
  center = FALSE,
  scale = FALSE,
  x = NULL,
  y = NULL
)
```

Arguments

`formula`	A formula.
`data`	A data.frame.
`center`	Logical; if enabled, centers the data and fits the intercept.
`scale`	Logical; if enabled, scales each feature by their standard deviations.
`x`	Design matrix.
`y`	Response matrix.

Value

An object of class `baritsu_blr`.

See Also

`mlpack::bayesian_linear_regression()` `predict.baritsu_blr()`

`linear_svm` *L2-regularized support vector machine*

Description

A wrapper around `mlpack::linear_svm()` that allows passing a formula.

Usage

```
linear_svm(
  formula = NULL,
  data = NULL,
  margin = 1,
  penalty = 1e-04,
  epochs = 1000,
  no_intercept = FALSE,
  tolerance = 1e-10,
  optimizer = c("lbfgs", "psgd"),
  stop_iter = 50,
  learn_rate = 0.01,
  shuffle = FALSE,
  seed = 0,
  x = NULL,
  y = NULL
)
```

Arguments

`formula`	A formula.
`data`	A data.frame.
`margin`	Margin of difference between correct class and other classes.
`penalty`	L2-regularization constant.
`epochs`	Maximum iterations for optimizer (0 indicates no limit). This argument is passed as `max_iterations`, not as `epochs` for `mlpack::linear_svm()`.
`no_intercept`	Logical; passed to `mlpack::linear_svm()`.
`tolerance`	Convergence tolerance for optimizer.
`optimizer`	Optimizer to use for training ("lbfgs" or "psgd").
`stop_iter`	Maximum number of full epochs over dataset for parallel SGD.
`learn_rate`	Step size for parallel SGD optimizer. in which data points are visited for parallel SGD.
`shuffle`	Logical; if true, doesn't shuffle the order.
`seed`	Random seed. If 0, `std::time(NULL)` is used internally.
`x`	Design matrix.
`y`	Response matrix.

Value

An object of class `baritsu_svm`.

See Also

`mlpack::linear_svm()` `predict.baritsu_svm()`

`logistic_regression` *L2-regularized logistic regression*

Description

A wrapper around `mlpack::logistic_regression()` that allows passing a formula.

Usage

```
logistic_regression(
  formula = NULL,
  data = NULL,
  penalty = 1e-04,
  epochs = 1000,
  decision_boundary = 0.5,
  tolerance = 1e-10,
  optimizer = c("lbfgs", "sgd"),
  batch_size = 64,
  learn_rate = 0.01,
  x = NULL,
  y = NULL
)
```

Arguments

`formula`	A formula.
`data`	A data.frame.
`penalty`	L2-regularization constant.
`epochs`	Maximum number of iterations.
`decision_boundary`	Decision boundary for prediction; if the logistic function for a point is less than the boundary, the class is taken to be 0; otherwise, the class is 1.
`tolerance`	Convergence tolerance for optimizer.
`optimizer`	Optimizer to use for training ("lbfgs" or "sgd").
`batch_size`	Batch size for SGD.
`learn_rate`	Step size for SGD optimizer.
`x`	Design matrix.
`y`	Response matrix.

Value

An object of class `baritsu_lgr`.

See Also

`mlpack::logistic_regression()` `predict.baritsu_lgr()`

`naive_bayes` *Parametric naive Bayes classifier*

Description

A wrapper around `mlpack::nbc()` that allows passing a formula.

Usage

```
naive_bayes(
  formula = NULL,
  data = NULL,
  incremental_variance = FALSE,
  x = NULL,
  y = NULL
)
```

Arguments

`formula`	A formula.
`data`	A data.frame.
`incremental_variance`	Logical; passed to `mlpack::nbc()`.
`x`	Design matrix.
`y`	Response matrix.

Value

An object of class `baritsu_nbc`.

See Also

`mlpack::nbc()` `predict.baritsu_nbc()`

perceptron *Single level neural network*

Description

A wrapper around `mlpack::perceptron()` that allows passing a formula.

Usage

```
perceptron(formula = NULL, data = NULL, epochs = 100, x = NULL, y = NULL)
```

Arguments

formula	A formula.
data	A data.frame.
epochs	Maximum number of iterations.
x	Design matrix.
y	Response matrix.

Value

An object of class `baritsu_prc`.

See Also

`mlpack::perceptron()` `predict.baritsu_prc()`

predict.baritsu *Prediction using mlpack via baritsu*

Description

Predicts with new data using a stored mlpack model.

Usage

```
## S3 method for class 'baritsu_ab'
predict(object, newdata, type = c("both", "class", "prob"), ...)

## S3 method for class 'baritsu_dt'
predict(object, newdata, type = c("both", "class", "prob"), ...)

## S3 method for class 'baritsu_ht'
predict(object, newdata, ...)
```

```
## S3 method for class 'baritsu_blr'
predict(object, newdata, ...)

## S3 method for class 'baritsu_lr'
predict(object, newdata, ...)

## S3 method for class 'baritsu_lgr'
predict(object, newdata, type = c("both", "class", "prob"), ...)

## S3 method for class 'baritsu_prc'
predict(object, newdata, ...)

## S3 method for class 'baritsu_sr'
predict(object, newdata, type = c("both", "class", "prob"), ...)

## S3 method for class 'baritsu_nbc'
predict(object, newdata, type = c("both", "class", "prob"), ...)

## S3 method for class 'baritsu_rf'
predict(object, newdata, type = c("both", "class", "prob"), ...)

## S3 method for class 'baritsu_svm'
predict(object, newdata, type = c("both", "class", "prob"), ...)
```

Arguments

`object`	An object out of baritsu function.
`newdata`	A data.frame.
`type`	Type of prediction. One of "both", "class", or "prob".
`...`	Not used.

Value

A tibble that contains the predictions and/or probabilities (and also the standard deviations of the predictive distribution only for `predict.baritsu_blr`).

`random_forest` *Random forests*

Description

A wrapper around `mlpack::random_forest()` that allows passing a formula.

Usage

```
random_forest(
  formula = NULL,
  data = NULL,
  mtry = 0,
  trees = 10,
  min_n = 1,
  maximum_depth = 0,
  minimum_gain_split = 0,
  seed = 0,
  x = NULL,
  y = NULL
)
```

Arguments

formula	A formula.
data	A data.frame.
mtry	Subspace dimension. If 0, autoselects the square root of data dimensionality.
trees	Number of trees.
min_n	Minimum number of data points in a leaf.
maximum_depth	Maximum depth of the tree.
minimum_gain_split	Minimum gain required to split an internal node.
seed	Random seed. If 0, `std::time(NULL)` is used internally.
x	Design matrix.
y	Response matrix.

Value

An object of class `baritsu_rf`.

See Also

`mlpack::random_forest()` `predict.baritsu_rf()`

---

softmax_regression *Softmax regression*

Description

A wrapper around `mlpack::softmax_regression()` that allows passing a formula.

Usage

```
softmax_regression(
  formula = NULL,
  data = NULL,
  penalty = 0.001,
  epochs = 400,
  no_intercept = FALSE,
  x = NULL,
  y = NULL
)
```

Arguments

formula	A formula.
data	A data.frame.
penalty	L2-regularization constant.
epochs	Maximum number of iterations.
no_intercept	Logical; passed to `mlpack::softmax_regression()`.
x	Design matrix.
y	Response matrix.

Value

An object of class `baritsu_sr`.

See Also

`mlpack::softmax_regression()` `predict.baritsu_sr()`

Index